Triangular NURBS and their Dynamic Generalizations

Hong Qin[1] and **Demetri Terzopoulos**
Department of Computer Science, University of Toronto,
10 King's College Road, Toronto, Ontario, M5S 1A4
E-mail addresses: qin@cs.toronto.edu; dt@cs.toronto.edu

Published in *Computer-Aided Geometric Design*, **??**(??):??–12??, 1996, in press.

Abstract

Triangular B-splines are a new tool for modeling a broad class of objects defined over arbitrary, non-rectangular domains. They provide an elegant and unified representation scheme for all piecewise continuous polynomial surfaces over planar triangulations. To enhance the power of this model, we propose triangular NURBS, the rational generalization of triangular B-splines, with weights as additional degrees of freedom. Fixing the weights to unity reduces triangular NURBS to triangular B-splines. Conventional geometric design with triangular NURBS can be laborious, since the user must manually adjust the many control points and weights. To ameliorate the design process, we develop a new model based on the elegant triangular NURBS geometry and principles of physical dynamics. Our model combines the geometric features of triangular NURBS with the demonstrated conveniences of interaction within a physics-based framework. The dynamic behavior of the model results from the numerical integration of differential equations of motion that govern the temporal evolution of control points and weights in response to applied forces and constraints. This results in physically meaningful hence highly intuitive shape variation. We apply Lagrangian mechanics to formulate the equations of motion of dynamic triangular NURBS and finite element analysis to reduce these equations to efficient numerical algorithms. We demonstrate several applications, including direct manipulation and interactive sculpting through force-based tools, the fitting of unorganized data, and solid rounding with geometric and physical constraints.

Keywords: CAGD, Physics-Based Modeling, Triangular NURBS, Triangular B-splines, Dynamics, Constraints, Finite Elements, Solid Rounding, Scattered Data Fitting, Interactive Sculpting.

[1]The author is now affiliated with:
Department of Computer & Information Science & Engineering
University of Florida

1 Introduction

Non-uniform rational B-splines, or NURBS, have become an industry standard by virtue of their many nice properties, such as their ability to represent free-form shapes as well as standard analytic shapes. However, the main drawback of tensor-product NURBS surfaces in solid modeling is that they are "topologically" rectangular. Consequently, the designer is forced to model multi-sided irregular shapes using degenerate patches with deteriorated inter-patch continuity. To compensate, explicit linear and/or non-linear smoothness constraints must be enforced on the patches, thus complicating the design task.

Triangular B-splines are emerging as a powerful new tool for solid modeling because they can represent, without degeneracy, complex objects defined on irregular parametric domains [7]. Using triangular B-splines, designers can represent shapes over triangulated planar domains with low degree piecewise polynomials that nonetheless maintain high-order continuity.[1] In addition, triangular B-splines offer a considerable breadth of geometric coverage. They subsume Bernstein-Bézier triangles with n-fold knots. Moreover, any piecewise polynomial surface over a planar triangulation may be represented as a linear combination of triangular B-splines [24]. Thus, triangular B-splines can also serve as a common representation for product data exchange and representation conversion.

In this paper we enhance the power of triangular spline models by proposing triangular NURBS, the rational generalization of triangular B-splines, with weights as additional degrees of freedom. As in conventional NURBS, fixing the weights to unity reduces triangular NURBS to triangular B-splines. Although triangular NURBS enable designers to overcome the limitations of tensor product NURBS, conventional geometric design with NURBS models can be problematic for the following reasons [28]:

- Normally the designer controls the geometry by assigning knots, positioning control points, and adjusting weights. Despite modern interaction technology, this "indirect," geometric degree of freedom oriented design process can be especially laborious for triangular splines because of the irregularity of control points and knot vectors.

- Design requirements are usually specified in terms of shape, not in terms of the geometric degrees of freedom of any particular shape representation. Because of the geometric "redundancy" of rational models,[2] indirect shape refinement can be *ad hoc* and ambiguous.

- Typical design requirements may be posed in both quantitative and qualitative terms. Therefore, it can be very frustrating to design via the indirect approach, say, a "fair" surface that approximates unorganized 3D data.

To ameliorate the design process for triangular NURBS, we develop a physics-based generalization of the model using Lagrangian mechanics and finite element techniques. Our new surface model, *dynamic triangular NURBS*, combines the elegant geometric features of triangular NURBS with the demonstrated conveniences of interaction within a physical dynamics framework. The following are some advantages of physics-based shape design [28]:

- Shape design is generally a time-varying process—the designer is often interested not only in the final shape but also in the intermediate shape variation. The behavior of our physics-based models result from the numerical integration of differential equations of (nonrigid) motion which automatically govern the temporal evolution of control points and weights in response to applied forces and constraints. This results in physically meaningful hence highly intuitive shape variation.

- Shapes can be sculpted in a direct manner using a variety of force-based "tools." Furthermore, functional design requirements can be readily implemented as deformation (fairness) energies and

[1]For example, quadratic triangular B-splines can yield C^1 continuous surfaces, whereas biquadratic tensor-product B-splines are necessary to achieve the same continuity.

[2]A particular shape can often be represented nonuniquely, with different values of knots, control points, and weights.

geometric constraints. In particular, as a dynamic model reaches equilibrium, it can serve as a nonlinear shape optimizer subject to the imposed constraints.[3]

- The physical model is built upon a standard geometric foundation. While shape design may proceed interactively or automatically at the physical level, existing geometric toolkits are concurrently applicable at the underlying geometric level.

Like their tensor product dynamic NURBS (D-NURBS) progenitors [28], dynamic triangular NURBS (or triangular D-NURBS) are a free-form, rational model that provides a systematic and unified technique for a variety of modeling tasks. We demonstrate several applications, including direct manipulation and interactive sculpting through forces and physical parameters, the fitting of unorganized 3D data, and solid rounding with geometric and physical constraints.

1.1 Background

The theoretical foundation of triangular B-splines lies in the multivariate simplex spline of approximation theory. Motivated by an idea of Curry and Schoenberg for a geometric interpretation of univariate B-splines, deBoor [8] first presented a brief description of multivariate simplex splines. Since then, their theory has been explored extensively [16, 5, 6, 13]. The well-known recurrence relation of multivariate simplex splines was introduced in [16]. Subsequently, Grandine [11] devised a stable evaluation algorithm. Dahmen and Micchelli [6] presented a thorough review of multivariate B-splines. From the point of view of blossoming, Dahmen, Micchelli and Seidel [7] proposed triangular B-splines which are essentially normalized simplex splines.

In contrast to the theory, the application of multivariate simplex splines has not been explored extensively, because of the complicated domain partitionings that may be required and the time-consuming evaluation and derivative computation algorithms, especially for high dimensional and high order cases. Fortunately, it is possible to derive efficient algorithms for a low dimensional domain such as a plane and/or a low order polynomial such as a quadratic or a cubic. Traas [29] discussed the applicability of bivariate quadratic simplex splines as finite elements and derived differentiation and inner product formulas. Auerbach et al. [1] use bivariate simplex B-splines to fit geological surfaces to scattered data by adjusting the triangulation of the parametric domain in accordance with the data distribution. In 1993, the first experimental CAGD software based on the triangular B-spline was developed, demonstrating the practical feasibility of multivariate B-spline algorithms [9]. Recently, Pfeifle and Seidel [19] demonstrate the fitting of triangular B-spline surfaces to scattered data through the use of least squares and optimization techniques.

A different area of research that provides background for our work in this paper is physics-based shape modeling. Terzopoulos and Fleischer [27] constructed free-form surfaces with natural dynamic behavior governed by the physical laws of elasticity and demonstrated simple interactive sculpting using viscoelastic and plastic models. Bloor and Wilson [3] demonstrated free-form design using tensor product B-splines and the optimization of energy functionals. Celniker and Gossard [4] developed an interesting prototype system for interactive design based on surface finite elements. Welch and Witkin [30] made similar use of trimmed hierarchical B-splines. Moreton and Sequin [18] interpolated a minimum energy curve network with quintic Bezier patches by minimizing the variation of curvature. In our earlier work, we developed dynamic NURBS (D-NURBS) [28, 23, 22], a physics-based generalization of standard geometric NURBS with full second-order dynamics. We demonstrated that D-NURBS allow a designer to interactively sculpt and directly manipulate shapes in a natural and predictable way using a variety of force-based tools and constraints.

[3]For example, appropriate NURBS weight values may be determined automatically subject to the constraints.

1.2 Overview

Section 2 reviews both multivariate simplex splines and triangular B-splines and proposes triangular NURBS. In Section 3, we formulate dynamic triangular NURBS and derive their equations of motion. Section 4 applies finite element analysis towards the numerical simulation of these equations. We discuss the physics-based design paradigm involving the use of forces and constraints in Section 5. Section 6 presents applications of dynamic triangular splines to interactive sculpting, scattered data fitting, and shape rounding. Section 7 concludes the paper.

2 Triangular NURBS Geometry

In this section we review the formulation of multivariate simplex splines and triangular B-splines, summarize their analytic and geometric properties, and straightforwardly generalize to the rational case—triangular NURBS.

2.1 Multivariate Simplex Splines

The basis functions of multivariate simplex splines may be defined either analytically or recursively [7, 16]. An s-variate simplex spline $M(\mathbf{x}|\{\mathbf{x}_0,\ldots,\mathbf{x}_m\})$ can be defined as a function of $\mathbf{x} \in R^s$ over the convex hull of a point set $\{\mathbf{x}_0,\ldots,\mathbf{x}_m\}$, depending on the $m+1$ knots $\mathbf{x}_i \in R^s$, $i=0,\ldots,m$, $(m \geq s)$. It is a piecewise polynomial with degree $d = m - s$ satisfying

$$\int_{R^s} f(\mathbf{x})M(\mathbf{x}|\{\mathbf{x}_0,\ldots,\mathbf{x}_m\})\,d\mathbf{x} = (m-s)!\int_{S^m} f(t_0\mathbf{x}_0+\ldots+t_m\mathbf{x}_m)\,dt_1\ldots dt_m, \tag{1}$$

where f is an arbitrary integrable function defined over the region which covers the convex hull spanned by the knot sequences $\mathbf{x}_i$. The region S^m is the standard m-simplex:

$$S^m = \{(t_1,\ldots,t_m)|\sum_{i=0}^{m} t_i = 1, t_i \geq 0\}.$$

$M(\mathbf{x}|\{\mathbf{x}_0,\ldots,\mathbf{x}_m\})$ has a very simple geometric interpretation: It is the projection of a higher dimensional simplex on a lower dimensional space. Let Δ be a m-simplex extended by $m+1$ vertices $[\mathbf{v}_0,\ldots,\mathbf{v}_m], \mathbf{v}_i \in R^m$ such that the projection of $\mathbf{v}_i$ on subspace R^s is $\mathbf{x}_i$, and for arbitrary $\mathbf{x}$ define a point set

$$A_{\mathbf{x}} = \{\mathbf{v}|\mathbf{v} \in \Delta, \mathbf{v}|_{R^s} = \mathbf{x}\}.$$

Then, we can explicitly formulate the basis function of s-variate simplex splines as

$$M(\mathbf{x}|\{\mathbf{x}_0,\ldots,\mathbf{x}_m\}) = \frac{(m-s)!}{m!}\cdot\frac{\mathrm{vol}_{m-s}(A_{\mathbf{x}})}{\mathrm{vol}_m(\Delta)}, \tag{2}$$

where vol_k denotes the k-dimensional volume of certain sets.

The basis function of multivariate simplex splines may also be formulated recursively, which facilitates evaluation and derivative computation: When $m = s$,

$$M(\mathbf{x}|\{\mathbf{x}_0,\ldots,\mathbf{x}_m\}) = \begin{cases} \frac{1}{m!\,\mathrm{vol}_s([\mathbf{x}_0,\ldots,\mathbf{x}_m])}, & x \in [\mathbf{x}_0,\ldots,\mathbf{x}_m] \\ 0 & \text{otherwise} \end{cases}$$

and when $m > s$,

$$M(\mathbf{x}|\{\mathbf{x}_0,\ldots,\mathbf{x}_m\}) = \sum_{i=0}^{m} \lambda_i M(\mathbf{x}|\{\mathbf{x}_0,\ldots,\mathbf{x}_{i-1},\mathbf{x}_{i+1},\ldots,\mathbf{x}_m\}), \tag{3}$$

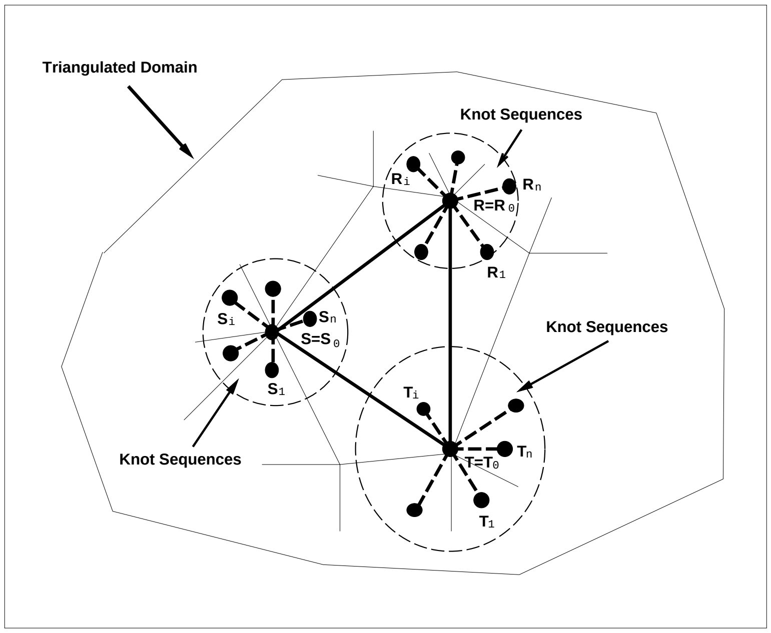


Figure 1: Knot vectors associated with each triangle in the domain triangulation.

where

$$\sum_{i=0}^{m} \lambda_i = 1; \qquad \sum_{i=0}^{m} \lambda_i \mathbf{x}_i = \mathbf{x}.$$

Note that when $m = s$ the basis function is discontinuous along the boundary of the convex hull $[\mathbf{x}_0, \ldots, \mathbf{x}_m]$; thus, the function value is not unique. Extra effort is therefore needed to deal with the boundary evaluation (see [9] for the concept of semi-open convex hull in the context of bivariate B-splines). Second, when $m > s$ the barycentric coefficients are not unique. An efficient method frequently used in applications is to make at least $m - s$ of the λ_i vanish, while the remaining ones are taken as positive barycentric coordinates to obtain stable and fast evaluation. Similarly, the directional derivative $D_{\mathbf{w}}$ of multivariate simplex splines may be recursively formulated as

$$D_{\mathbf{w}} M(\mathbf{x}|\{\mathbf{x}_0, \ldots, \mathbf{x}_m\}) = \mathbf{w}^{\top} \nabla M = (m - s) \sum_{i=0}^{m} \mu_i M(x|\{\mathbf{x}_0, \ldots, \mathbf{x}_{i-1}, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_m\}), \tag{4}$$

where μ_i are coefficients satisfying

$$\sum_{i=0}^{m} \mu_i = 0; \qquad \sum_{i=0}^{m} \mu_i \mathbf{x}_i = \mathbf{w}.$$

Again, these scalar coefficients are not unique. An efficient algorithm to evaluate derivatives is obtained by setting $\mu_i = D_{\mathbf{w}} \lambda_i$, $i = 0, \ldots, m$, where λ_i is defined in (3).

2.2 Triangular NURBS

The triangular B-spline is essentially a normalized simplex spline. Let $T = \{\Delta(\mathbf{i}) = [\mathbf{r}, \mathbf{s}, \mathbf{t}] | \mathbf{i} = (i_0, i_1, i_2) \in Z_+^3\}$ be an arbitrary triangulation of the planar parametric domain, where i_0, i_1, and i_2 denote indices of $\mathbf{r}$, $\mathbf{s}$, and $\mathbf{t}$ in the vertex array of the triangulation, respectively. For each vertex $\mathbf{v}$ in the triangulated domain, we then assign a knot sequence (also called a cloud of knots) $[\mathbf{v} = \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_n]$ (which are inside the shaded circles in Fig. 1). Next, we define a convex hull

$$V_{\mathbf{i},\beta} = \{\mathbf{r}_0, \ldots, \mathbf{r}_{\beta_0}, \mathbf{s}_0, \ldots, \mathbf{s}_{\beta_1}, \mathbf{t}_0, \ldots, \mathbf{t}_{\beta_2}\},$$

where subscript **i** is a triangle index and $\beta = (\beta_0, \beta_1, \beta_2)$ is a triplet such that $|\beta| = \beta_0 + \beta_1 + \beta_2 = n$. The bivariate simplex spline $M(\mathbf{u}|V_{\mathbf{i},\beta})$ with degree n over $V_{\mathbf{i},\beta}$ can be defined recursively (see [7] for the details), where $\mathbf{u} = (u, v)$ defines the triangulated parametric domain of the surface. We then define a bivariate B-spline basis function as

$$N_{\mathbf{i},\beta}(\mathbf{u}) = |d(\mathbf{r}_{\beta_0}, \mathbf{s}_{\beta_1}, \mathbf{t}_{\beta_2})| M(\mathbf{u}|V_{\mathbf{i},\beta}), \tag{5}$$

where $|d(\mathbf{r}_{\beta_0}, \mathbf{s}_{\beta_1}, \mathbf{t}_{\beta_2})|$ is twice the area of $\Delta(\mathbf{r}_{\beta_0}, \mathbf{s}_{\beta_1}, \mathbf{t}_{\beta_2})$. Like the ordinary tensor-product B-spline, a triangular B-spline surface of degree n over arbitrary triangulated domain is the combination of a set of basis functions with control points $\mathbf{p}_{\mathbf{i},\beta}$:

$$\mathbf{s}(\mathbf{u}) = \sum_{\mathbf{i}} \sum_{|\beta|=n} \mathbf{p}_{\mathbf{i},\beta} N_{\mathbf{i},\beta}(\mathbf{u}). \tag{6}$$

Generalizing (6) by associating a weight $w_{\mathbf{i},\beta}$ with each control point, we define triangular NURBS as the combination of a set of piecewise rational functions:

$$\mathbf{s}(\mathbf{u}) = \frac{\sum_{\mathbf{i}} \sum_{|\beta|=n} \mathbf{p}_{\mathbf{i},\beta} w_{\mathbf{i},\beta} N_{\mathbf{i},\beta}(\mathbf{u})}{\sum_{\mathbf{i}} \sum_{|\beta|=n} w_{\mathbf{i},\beta} N_{\mathbf{i},\beta}(\mathbf{u})}. \tag{7}$$

2.3 Properties

Like non-rational B-splines, the rational nonnegative basis functions of triangular NURBS sum to unity. They inherit many of the properties of non-rational B-splines, such as the convex hull property, local support, affine invariance, and form a common representation for any piecewise polynomial [7, 25, 9, 12]. Moreover, they have some additional properties:

- Triangular NURBS and their rational basis functions are infinitely smooth in the interior of non-overlapping sub-triangles formed by the knot nets, provided the denominator is nonzero. At the boundary of sub-triangles, they are C^{n-1} continuous if the knots are in general position. The designer can obtain different smoothness conditions by varying the knot arrangement.

- Triangular NURBS include weights as extra degrees of freedom which influence local shape. If a particular weight is zero, then the corresponding rational basis function is also zero and its control point does not effect the NURBS shape. The spline is "attracted" toward a control point more if the corresponding weight is increased and less if the weight is decreased.

Shape design based on triangular NURBS includes the specification of a domain triangulation, knot sequences, and a control polygon to generate an initial shape. The initial shape is then refined into the final desired shape through interactive adjustment of control points, weights, and knots. The availability of weights as additional degrees of freedom expands the geometric coverage of triangular NURBS. In contrast to the case of regular NURBS, however, the special analytic shapes that can be represented precisely by triangular NURBS remains an open question. Because of the irregularity of the triangulation vertices and knot sequences, the shape refinement process is *ad hoc* and it can become extremely tedious. Hence, the considerable geometric flexibility of triangular NURBS can thwart the conventional geometric design approach. To improve matters, we propose a physics-based triangular NURBS model.

3 Physics-Based Triangular NURBS

In this section, we formulate a dynamic model based on triangular NURBS. The control points and weights of the geometric model of section 2 become generalized (physical) coordinates in the dynamic model. We derive the Jacobian and basis function matrices that lead to compact expressions for the velocity and position functions of the surface. We introduce time, mass, and deformation energy into the triangular NURBS formulation and employ Lagrangian dynamics to derive the equations of motion of the physics-based model.

3.1 Geometry and Kinematics

The dynamic triangular NURBS extend the geometric triangular NURBS in (7) by explicitly incorporating time and physical behavior. The surface is a function of both the parametric variable $\mathbf{u}$ and time t:

$$\mathbf{s}(\mathbf{u}, t) = \frac{\sum_{\mathbf{i}} \sum_{|\beta|=n} \mathbf{p}_{\mathbf{i},\beta}(t) w_{\mathbf{i},\beta}(t) N_{\mathbf{i},\beta}(\mathbf{u})}{\sum_{\mathbf{i}} \sum_{|\beta|=n} w_{\mathbf{i},\beta}(t) N_{\mathbf{i},\beta}(\mathbf{u})}. \tag{8}$$

To simplify notation, we define the vector of generalized coordinates (control points) $\mathbf{p}_{\mathbf{i},\beta}$ and (weights) $w_{\mathbf{i},\beta}$ as

$$\mathbf{p} = [\ldots, \mathbf{p}_{\mathbf{i},\beta}^{\top}, w_{\mathbf{i},\beta}, \ldots]^{\top}.$$

We then express (8) as $\mathbf{s}(\mathbf{u}, \mathbf{p})$ in order to emphasize its dependence on $\mathbf{p}$ whose components are functions of time.

Thus, the velocity of the dynamic triangular NURBS is

$$\dot{\mathbf{s}}(\mathbf{u}, \mathbf{p}) = \mathbf{J}\dot{\mathbf{p}}, \tag{9}$$

where the overstruck dot denotes a time derivative and the Jacobian matrix $\mathbf{J}(\mathbf{u}, \mathbf{p})$ is the concatenation of the vectors $\partial\mathbf{s}/\partial\mathbf{p}_{\mathbf{i},\beta}$ and $\partial\mathbf{s}/\partial w_{\mathbf{i},\beta}$. Assuming m triangles in the parametric domain, β traverses $k = (n+2)!/(n!2!)$ possible triplets whose components sum to n. Because $\mathbf{s}$ is a 3-vector and $\mathbf{p}$ is an $M = 4mk$ dimensional vector, $\mathbf{J}$ is a $3 \times M$ matrix, which may be written as

$$\mathbf{J} = \left[\ldots, \begin{bmatrix} R_{\mathbf{i},\beta} & 0 & 0 \\ 0 & R_{\mathbf{i},\beta} & 0 \\ 0 & 0 & R_{\mathbf{i},\beta} \end{bmatrix}, \mathbf{w}_{\mathbf{i},\beta}, \ldots \right] \tag{10}$$

where

$$R_{\mathbf{i},\beta}(\mathbf{u}, \mathbf{p}) = \frac{\partial \mathbf{s}_x}{\partial \mathbf{p}_{\mathbf{i},\beta,x}} = \frac{\partial \mathbf{s}_y}{\partial \mathbf{p}_{\mathbf{i},\beta,y}} = \frac{\partial \mathbf{s}_z}{\partial \mathbf{p}_{\mathbf{i},\beta,z}} = \frac{w_{\mathbf{i},\beta} N_{\mathbf{i},\beta}(\mathbf{u})}{\sum_{\mathbf{j}} \sum_{|\alpha|=n} w_{\mathbf{j},\alpha} N_{\mathbf{j},\alpha}(\mathbf{u})}$$

and

$$\mathbf{w}_{\mathbf{i},\beta}(\mathbf{u}, \mathbf{p}) = \frac{\partial \mathbf{s}}{\partial w_{\mathbf{i},\beta}} = \frac{(\mathbf{p}_{\mathbf{i},\beta} - \mathbf{s}) N_{\mathbf{i},\beta}(\mathbf{u})}{\sum_{\mathbf{j}} \sum_{|\alpha|=n} w_{\mathbf{j},\alpha} N_{\mathbf{j},\alpha}(\mathbf{u})}$$

The subscripts x, y, and z denote derivatives of the components of a 3-vector. Moreover, we can express the surface as the product of the Jacobian matrix and the generalized coordinate vector:

$$\mathbf{s}(\mathbf{u}, \mathbf{p}) = \mathbf{J}\mathbf{p}. \tag{11}$$

The proof of (11) is the same as that for D-NURBS in [28].

3.1.1 Lagrange Equations of Motion

We derive the equations of motion of our dynamic triangular NURBS by applying Lagrangian dynamics [10]. We express the kinetic energy due to a prescribed mass distribution function $\mu(u, v)$ over the parametric domain of the surface and a dissipation energy due to a damping density function $\gamma(u, v)$. To define an elastic potential energy, we adopt the *thin-plate under tension* energy model [26, 4, 30]

$$U = \frac{1}{2} \iint \left(\alpha_{1,1} \mathbf{s}_u^2 + \alpha_{2,2} \mathbf{s}_v^2 + \beta_{1,1} \mathbf{s}_{uu}^2 + \beta_{1,2} \mathbf{s}_{uv}^2 + \beta_{2,2} \mathbf{s}_{vv}^2 \right) du\, dv.$$

The subscripts on $\mathbf{s}$ denote parametric partial derivatives. The $\alpha_{i,j}(u, v)$ and $\beta_{i,j}(u, v)$ are elasticity functions which control tension and rigidity, respectively. Other energies are applicable, including, at greater computational cost, the nonquadratic, curvature-based energies in [27, 18]. Applying the Lagrangian formulation, we obtain the second-order nonlinear equations of motion

$$\mathbf{M}\ddot{\mathbf{p}} + \mathbf{D}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}_p + \mathbf{g}_p, \tag{12}$$

where the mass matrix is

$$\mathbf{M}(\mathbf{p}) = \iint \mu \mathbf{J}^\top \mathbf{J}\, du\, dv,$$

the damping matrix is

$$\mathbf{D}(\mathbf{p}) = \iint \gamma \mathbf{J}^\top \mathbf{J}\, du\, dv,$$

and the stiffness matrix is

$$\mathbf{K}(\mathbf{p}) = \iint (\alpha_{1,1} \mathbf{J}_u^\top \mathbf{J}_u + \alpha_{2,2} \mathbf{J}_v^\top \mathbf{J}_v + \beta_{1,1} \mathbf{J}_{uu}^\top \mathbf{J}_{uu} + \beta_{1,2} \mathbf{J}_{uv}^\top \mathbf{J}_{uv} + \beta_{2,2} \mathbf{J}_{vv}^\top \mathbf{J}_{vv})\, du\, dv.$$

All are $M \times M$ matrices. The generalized forces on generalized coordinates due to the applied force distribution $\mathbf{f}(u, v, t)$ is

$$\mathbf{f}_p(\mathbf{p}) = \iint \mathbf{J}^\top \mathbf{f}(u, v, t)\, du\, dv.$$

Because of the geometric nonlinearity, the generalized inertial forces

$$\mathbf{g}_p(\mathbf{p}) = -\iint \mu \mathbf{J}^\top \dot{\mathbf{J}} \dot{\mathbf{p}}\, du\, dv$$

appear in the equations of motion. The derivation of (12) proceeds as for D-NURBS (see [28] for the details).

4 Finite Element Implementation

The evolution of the generalized coordinates $\mathbf{p}(t)$, determined by (12) with time-varying matrices, cannot be solved analytically in general. Instead, we pursue an efficient numerical implementation using finite-element techniques [14]. Standard finite element codes explicitly assemble the global matrices that appear in the discrete equations of motion [14]. This approach would be too expensive for interactive performance. We use an iterative matrix solver to avoid the cost of assembling the global matrices **M**, **D**, and **K**, working instead with the individual element matrices. We construct finite element data structures that permit the parallel computation of the element matrices. The remainder of this section provides the details of our numerical implementation.

4.1 Discrete Dynamics Equations

To integrate (12) in an interactive modeling system, it is important to provide the designer with visual feedback about the evolving state of the dynamic model. Rather than using costly time integration methods that take the largest possible time steps, it is more crucial to support a smooth animation by maintaining the continuity of the dynamics from one step to the next. Hence, less costly yet stable time integration methods that take modest time steps are desirable.

The state of the dynamic triangular NURBS at time $t+\Delta t$ is integrated using prior states at time t and $t-\Delta t$. To maintain the stability of the integration scheme, we use an implicit time integration method, which employs discrete derivatives of $\mathbf{p}$ using backward differences

$$\ddot{\mathbf{p}}^{(t+\Delta t)} \approx (\mathbf{p}^{(t+\Delta t)} - 2\mathbf{p}^{(t)} + \mathbf{p}^{(t-\Delta t)})/\Delta t^2$$

and

$$\dot{\mathbf{p}}^{(t+\Delta t)} \approx (\mathbf{p}^{(t+\Delta t)} - \mathbf{p}^{(t-\Delta t)})/2\Delta t.$$

We obtain the time integration formula

$$\left(2\mathbf{M} + \Delta t\mathbf{D} + 2\Delta t^2\mathbf{K}\right)\mathbf{p}^{(t+\Delta t)} = 2\Delta t^2(\mathbf{f}_p + \mathbf{g}_p) + 4\mathbf{M}\mathbf{p}^{(t)} - (2\mathbf{M} - \Delta t\mathbf{D})\mathbf{p}^{(t-\Delta t)}, \tag{13}$$

where the superscripts denote evaluation of the quantities at the indicated times. The matrices and forces are evaluated at time t. We employ the conjugate gradient method [21] to obtain an iterative solution for $\mathbf{p}^{(t+\Delta t)}$ at each time step. To achieve interactive simulation rates, we limit the number of conjugate gradient iterations per time step to 10. More than 2 iterations tend to be necessary when the physical parameters are changed significantly during the numerical simulation. Hence, it is possible to achieve interactive simulation rates on common graphics workstations. We have observed that quadratic and cubic surfaces with about 100 control points may be simulated at real-time interactive rates.

It is possible to make simplifications that further reduce the computational expense of (13), making it practical to work with larger triangular NURBS surfaces. First, it is seldom necessary to simulate the fully general triangular NURBS model throughout an entire sculpting session. Once we freeze the values of the weights, all of the matrices in (12) are constant and their entries need no longer be recomputed at each time step. Interactive rates are readily obtained for surfaces with up to an order of magnitude more degrees of freedom with this restricted rational generalization of the triangular B-splines. Moreover, triangular NURBS reduce to dynamic B-splines if all the frozen weights are set equal to 1. Second, although (12) will generate realistic dynamics for physics-based graphics animation, in certain CAGD applications where the designer is interested only in the final equilibrium configuration of the model, we can simplify (12) by setting the mass density function $\mu(u, v)$ to zero, so that the inertial terms vanish. This economizes on storage and makes the algorithm more efficient. With zero mass density, (12) reduces to

$$\mathbf{D}\dot{\mathbf{p}} + \mathbf{K}\mathbf{p} = \mathbf{f}_p. \tag{14}$$

Discretizing the derivatives of $\mathbf{p}$ in (14) with backward differences, we obtain the integration formula

$$(\mathbf{D} + \Delta t\mathbf{K})\,\mathbf{p}^{(t+\Delta t)} = \Delta t\mathbf{f}_p + \mathbf{D}\mathbf{p}^{(t)}. \tag{15}$$

4.2 Element Data Structure

We construct triangular finite element data structures that permit the parallel computation of the element matrices. Fig. 2 illustrates a typical triangular spline finite element, along with its local degrees of freedom. Note that, the degrees of freedom of this finite element consist of all control points and weights whose basis functions are non-zero over the current triangle in the parametric domain. Also, the weights are not explicitly provided in Fig. 2. Because of the irregular knot distribution of triangular splines, we do not

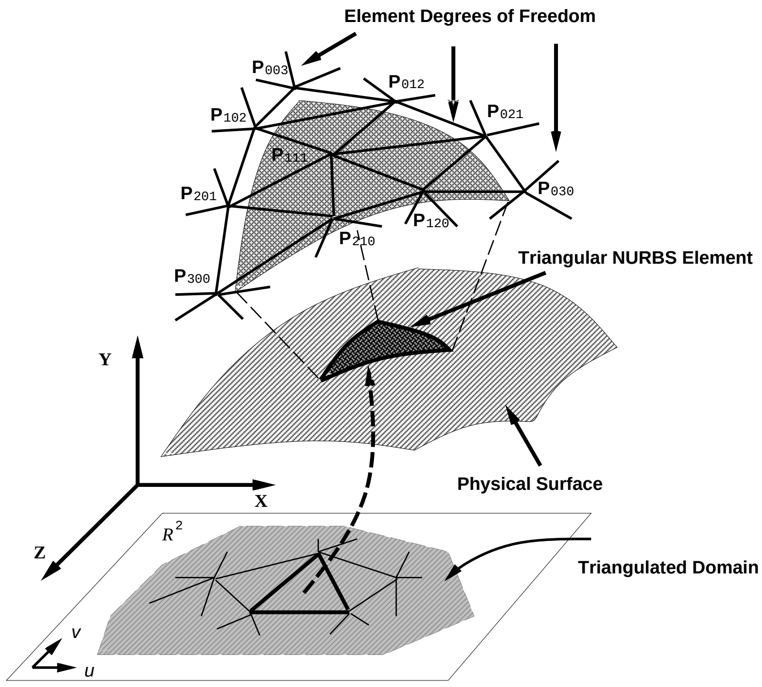


Figure 2: One finite element and its degrees of freedom of a triangular NURBS surface.

display all the degrees of freedom for this finite element; only 10 indexed control points are shown in Fig. 2.

We define an element data structure which contains the geometric specification of the triangular patch element along with its physical properties. In each element, we allocate an elemental mass, damping, and stiffness matrix, and include the physical quantities—the mass $\mu(u,v)$, damping $\gamma(u,v)$, and elasticity $\alpha_{i,j}(u,v)$, $\beta_{i,j}(u,v)$ density functions. A complete dynamic triangular NURBS model then consists of an ordered array of elements with additional information. The element structure includes pointers to appropriate components of the global vector of generalized coordinates $\mathbf{p}$ (control points and weights). Neighboring elements will share some generalized coordinates.

4.3 Calculation of Element Matrices

The integral expressions for the mass, damping, and stiffness matrices associated with each element are evaluated numerically using Gaussian quadrature [21]. We explain the computation of the element mass matrix; the computation of the damping and stiffness matrices follow suit. The expression for entry m_{ij} of the element mass matrix takes the integral form

$$m_{ij} = \iint_{\Delta(\mathbf{r},\mathbf{s},\mathbf{t})} \mu(u,v) f_{ij}(u,v)\, du\, dv,$$

where $\Delta(\mathbf{r},\mathbf{s},\mathbf{t})$, is the parametric domain of the element. Given integers N_g, we can find Gauss weights a_g and abscissas u_g and v_g in the two parametric directions such that m_{ij} can be approximated by

$$m_{ij} \approx \sum_{g=1}^{N_g} a_g \mu(u_g, v_g) f_{ij}(u_g, v_g).$$

We apply the recursive algorithm of multivariate simplex splines [16] to evaluate $f_{ij}(u_g, v_g)$. In our implementation we choose N_g to be 7 for quadratic and cubic triangular splines.

Note that because of the irregular knot distribution, the integrands f_{ij}'s may vanish over subregions of $\Delta(\mathbf{r},\mathbf{s},\mathbf{t})$. We can minimize numerical quadrature error by further subdividing the $\Delta(\mathbf{r},\mathbf{s},\mathbf{t})$. We subdivide

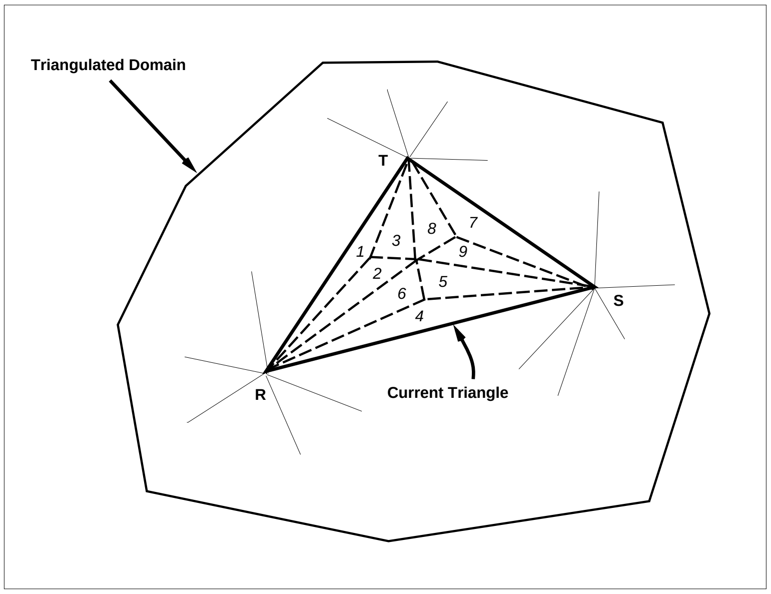


Figure 3: Nine subtriangles for numerical quadrature.

a triangular into 9 subtriangles as shown in Fig. 3 and observe that matrices computed according to this subdivision lead to stable, convergent solutions. Alternatively, a more precise albeit expensive approach is to convert a triangular B-spline into piecewise Bezier surfaces defined on a *finer* triangulation with extra knot lines (Fig. 4 illustrates the quadratic case). Note that Fig. 4 shows only a subset of the finer triangulation comprising the intersection of knot lines. The *complete* finer triangulation can be obtained by adding extra edges [9] into Fig. 4. This subdivision scheme yields about $10 - 100$ subtriangles per parametric domain triangle for cubics and quadratics.

5 Dynamic Interaction

The physics-based shape design approach allows modeling requirements to be expressed and satisfied through the use of energies, forces, and constraints. The designer may apply time-varying forces to sculpt shapes interactively or to optimally approximate data. Certain aesthetic constraints (such as "fairness") are expressible in terms of elastic energies that give rise to specific stiffness matrices $\mathbf{K}$. Other constraints include position or normal specification at surface points and continuity requirements between adjacent patches. By building the dynamic model upon the triangular NURBS geometry, we allow the designer to continue to use the repertoire of advanced geometric tools that have become prevalent, among them, the imposition of geometric constraints that the final shape must satisfy. Our physics-based shape design approach [28, 23, 22] which utilizes energies, forces, and constraints has proven to be simpler and more intuitive than conventional geometric design methods (e.g., the manipulation and adjustment of control points and weights). This approach is even more attractive for triangular NURBS, because of the complexity and irregularity of their control point and knot vectors.

5.1 Force Tools

Sculpting tools may be implemented as applied forces. The force distribution $\mathbf{f}(u, v, t)$ represents the net effect of all applied forces. Typical force functions are spring forces, repulsion forces, gravitational forces, inflation forces, etc. [27]. Consider connecting a material point (u_0, v_0) of a dynamic triangular NURBS surface to a point $\mathbf{d}_0$ in space with an ideal Hookean spring of stiffness k. The net applied spring force is

$$\mathbf{f}(u, v, t) = \iint k(\mathbf{d}_0 - \mathbf{s}(u, v, t))\delta(u - u_0, v - v_0)\, du\, dv, \tag{16}$$

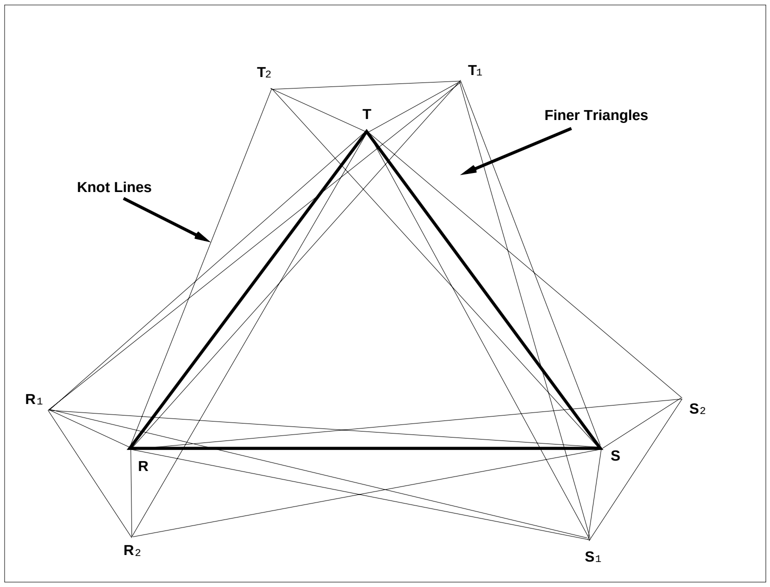


Figure 4: Finer triangulation due to intersection of knot lines (see text).

where the δ is the unit delta function. Equation (16) implies that $\mathbf{f}(u_0, v_0, t) = k(\mathbf{d}_0 - \mathbf{s}(u_0, v_0, t))$ and vanishes elsewhere on the surface, but we can generalize it by replacing the δ function with a smooth kernel (e.g., a unit Gaussian) to spread the applied force over a greater portion of the surface. In general, the points (u_0, v_0) and $\mathbf{d}_0$ need not be constant. We can control either or both using a mouse to obtain an interactive spring force. More advanced force tools are readily implemented to intuitively manipulate geometrically intrinsic quantities such as normal and curvature anywhere on the surface.

5.2 Constraints

In practical applications, design requirements may be posed as a set of physical parameters or as geometric constraints. Nonlinear constraints can be enforced through Lagrange multiplier techniques [17, 20]. This approach increases the number of degrees of freedom, hence the computational cost, by adding unknowns λ_i, known as Lagrange multipliers, which determine the magnitudes of the constraint forces. The augmented Lagrangian method [17] combines the Lagrange multipliers with the simpler penalty method. The Baumgarte stabilization method [2] solves constrained equations of motion through linear feedback control (see also [15, 28]). These techniques are appropriate for enforcing constraints on dynamic triangular NURBS.

Linear geometric constraints such as point, curve, and surface normal constraints can be easily incorporated into dynamic triangular NURBS by reducing the matrices and vectors in (12) to a minimal unconstrained set of generalized coordinates. For example, by confining all associated weights to be unity, we obtain dynamic triangular B-splines. Furthermore, we arrive at the *continuous net* [9], which is a special case of general triangular B-splines, by constraining respective control points along common boundaries of two adjacent triangles in parametric triangulation (Fig. 5). Linear constraints can be implemented by applying the same numerical solver on an unconstrained subset of $\mathbf{p}$. See [28] for a detailed discussion on linear constraints.

Rational dynamic models have an interesting peculiarity due to the weights. While the control point components of $\mathbf{p}$ may take arbitrary finite values in $\Re$, negative weights may cause the denominator to vanish at some evaluation points, causing the matrices to diverge. Although not forbidden, negative weights are not useful. We enforce the positivity of weights at each simulation time step by simply projecting any weight value that has drifted below a small positive threshold back to this lower bound. Alternatively, we can give the designer the option of constraining the weights near certain desired target values using

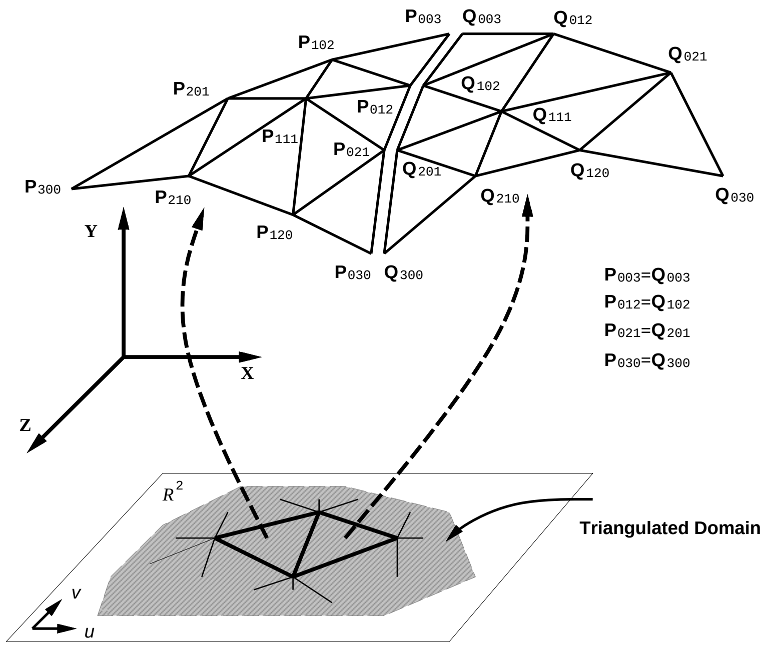


Figure 5: A constrained triangular B-Spline configuration: *continuous net.*

penalty methods [28].

6 Modeling Applications

We have developed prototype modeling software based on dynamic triangular B-splines (an advanced system based upon dynamic triangular NURBS is under construction). We have adopted the data structure, file, and rendering formats of existing geometric triangular B-spline software [9]. To implement the Lagrangian dynamics model on top of this software, we have had to implement a new algorithm for simultaneously evaluating non-zero basis functions and their derivatives up to second order at arbitrary domain points for finite element assembly and dynamic simulation. Our parallelized iterative numerical algorithm takes advantage of an SGI Iris 4D/380VGX shared memory multiprocessor.

In accordance with the physics-based paradigm, users can sculpt surface shapes by applying simulated forces. They can satisfy design requirements by adjusting the internal physical parameters such as the mass, damping, and stiffness densities, along with the force gain factors. In our prototype system, linear constraints such as the freezing of control points have been associated with physics-based toolkits. Local geometric constraints can be used to achieve real-time local manipulation for interactive sculpting of complex objects.

In the following sections we demonstrate applications of dynamic triangular B-splines to interactive sculpting, solid rounding, and scattered data fitting. Table 1 specifies the physical parameters used in the subsequent experiments. Fig. 6 illustrates the parametric domain triangulation of the various surfaces used in these experiments.

6.1 Rounding

The rounding operation is usually attempted geometrically by enforcing continuity requirements on the fillet which interpolates between two or more surfaces. By contrast, the dynamic triangular B-spline can produce a smooth fillet by minimizing its internal deformation energy subject to position and normal constraints. The dynamic simulation automatically produces the desired final shape as it achieves static equilibrium.

Applications	*Physical Parameters*								
	μ	γ	$\alpha_{1,1}$	$\alpha_{2,2}$	$\beta_{1,1}$	$\beta_{1,2}$	$\beta_{2,2}$	Δt	k
Edge rounding	0.0	500.0	1000.0	0.0	1.0	0.0	0.0	0.04	0.0
Corner rounding	0.0	50.0	1000.0	1000.0	10.0	10.0	10.0	0.04	0.0
Bevel rounding	0.0	25.0	100.0	100.0	0.0	0.0	0.0	0.04	0.0
Hill fitting	0.0	10.0	0.0	0.0	10.0	10.0	10.0	0.04	1000.0
Convex/Concave fitting	0.0	5.0	0.0	0.0	5.0	5.0	5.0	0.04	2000.0
Mountain/Valley fitting	0.0	25.0	0.0	0.0	1.0	1.0	1.0	0.04	2000.0
Quadratic object sculpting	5.0	25.0	10.0	10.0	1000.0	1000.0	1000.0	0.04	2000.0
Cubic patch sculpting	5.0	10.0	100.0	100.0	10.0	10.0	10.0	0.04	1000.0

Table 1: Physical parameters used in the examples. Parameter k denotes the stiffness of the spring force.

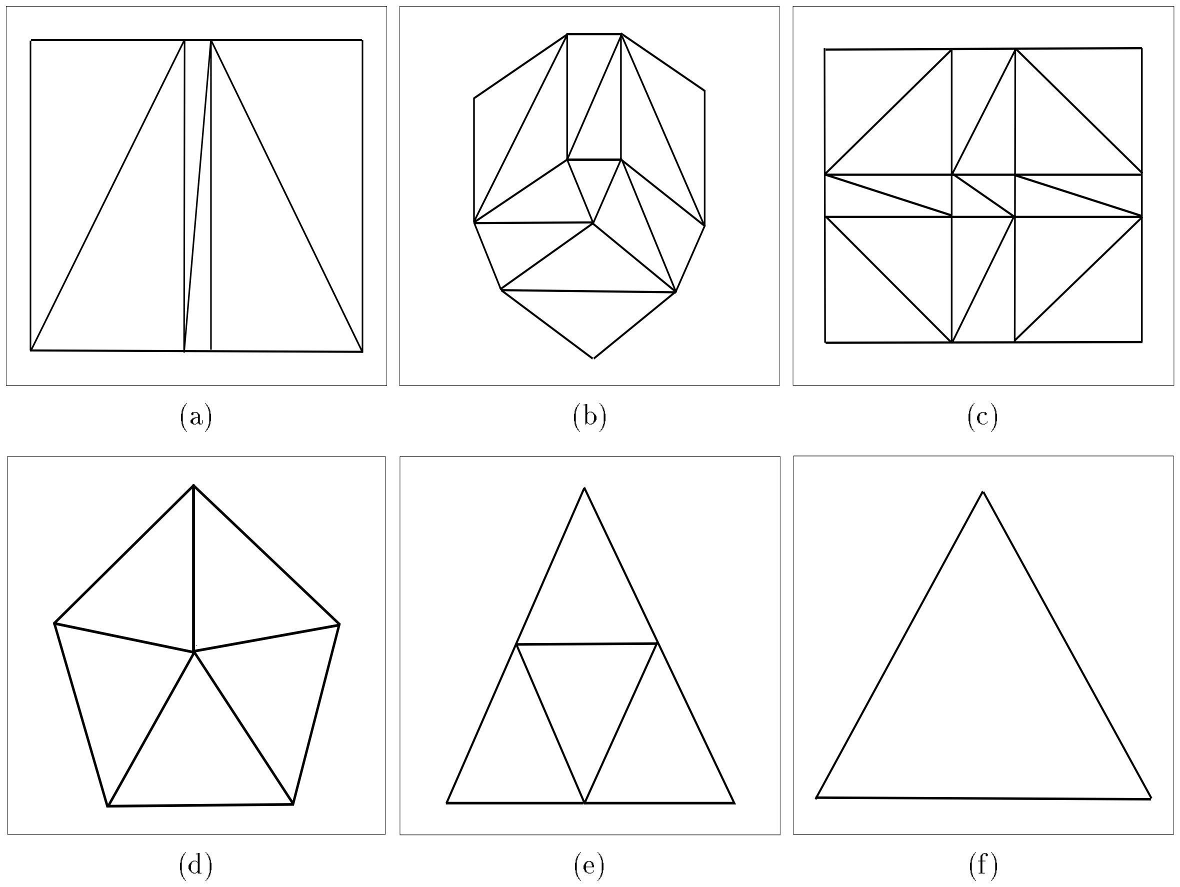


Figure 6: Domain triangulation of surfaces used in the examples (see text). (a) An edge. (b) A trihedral corner. (c) A bevel joint. (d) A pentagonal surface. (e) An open quadratic surface. (f) A cubic patch.

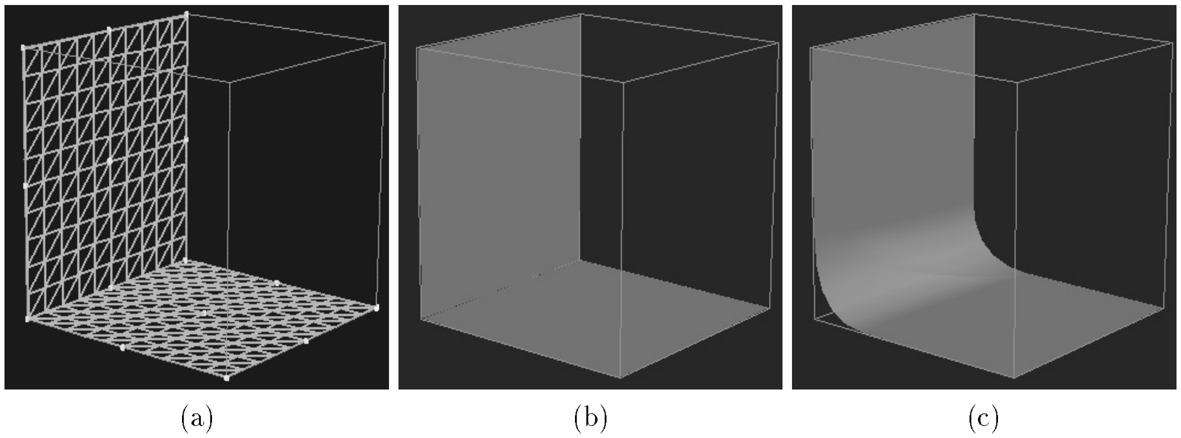


Figure 7: Rounding of an edge. (a) Initial wireframe object. (b) Initial shaded object. (c) Rounded object.

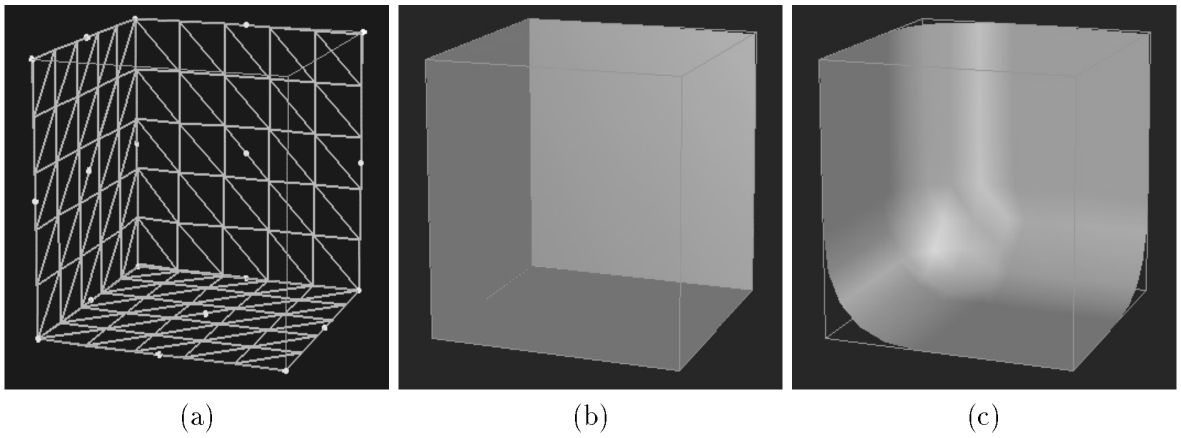


Figure 8: Rounding of a trihedral corner. (a) Initial wireframe object. (b) Initial shaded object. (c) Rounded object.

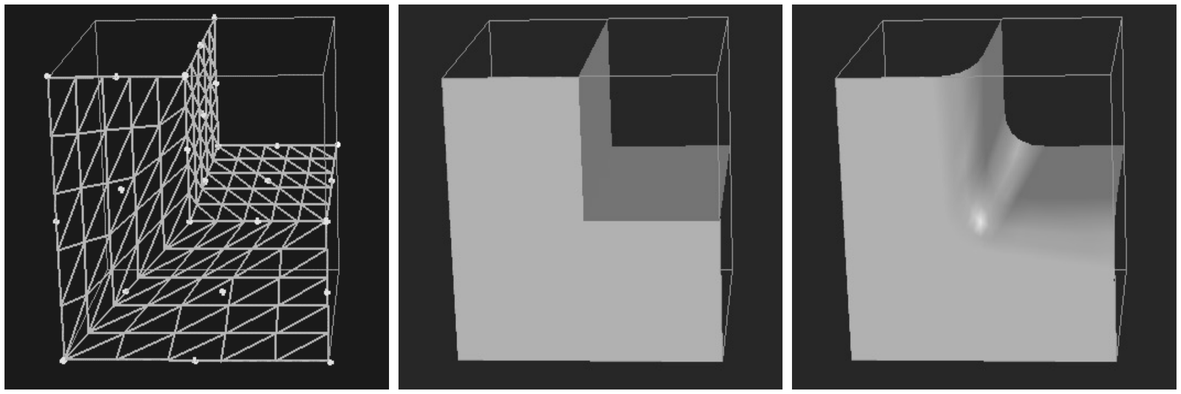
Fig. 7 demonstrates the rounding of a sharp edge represented by a quadratic triangular B-spline surface with 36 control points. The sharp edge can be represented exactly with multiple control points (note that a sharp edge can also be obtained with colinear knots). By restricting the control polygon to be a continuous net, we reduced the number of control points to 21. The initial wireframe and shaded shapes are shown in Fig. 7(a–b). After initiating the physical simulation, the sharp edges are rounded as the final shape equilibrates into the minimal energy state shown in Fig. 7(c).

Fig. 8 illustrates the rounding of a trihedral corner of a cube. The corner is represented using a quadratic triangular B-spline with 78 control points. The initial wireframe and shaded shapes are demonstrated in Fig. 8(a–b). The rounding operation is applied in the vicinity of three sharp edges. The sharp edges and corner are rounded with position and normal constraints along the far boundaries of the faces shown in Fig. 8(c).

Fig. 9 shows a rounding example involving a bevel joint. The bevel joint is a quadratic triangular B-spline with 108 control points. The initial right-angle joint and the final rounded shapes are shown in Fig. 9(a–c).

6.2 Scattered Data Fitting

A useful modeling technique generally known as scattered data fitting is based on fitting surfaces to unorganized point constraints. Interesting situations arise when there are fewer or more data points than



(a) (b) (c)

Figure 9: Rounding of a bevel joint. (a) Initial wireframe object. (b) Initial shaded object. (c) Rounded object.

there are degrees of freedom in the model, leading to underconstrained or overconstrained fitting problems, respectively. The inclusion of an elastic energy in our dynamic surfaces makes them applicable to such problems.

The data interpolation problem is amenable to common constraint techniques [17]. Approximation can be achieved by physically coupling the dynamic triangular B-splines to the data through Hookean spring forces (16). We interpret $\mathbf{d}_0$ in (16) as the data point (generally in $\Re^3$) and (u_0, v_0) as the parametric coordinates associated with the data point (typically the nearest material point of the surface). The spring constant c determines the closeness of fit to the data point.

We present three examples of surface fitting using dynamic triangular B-spline surfaces coupled to data points through spring forces. The initial surface is a quadratic pentagon with 30 control points defined over the pentagonal domain. Three different sets of 6 data points are shown in Fig. 10(a1–c1) along with the initial surfaces. The spring forces associated with the data points are applied to the nearest points on the surfaces. Note that the spring attachments shown in Fig. 10(a1–c1) show the initial correspondence and are not fixed during the dynamic surface fitting process. Fig. 10(a2–c2) show the final fitted surfaces.

6.3 Dynamic Interactive Sculpting

The physics-based modeling approach is ideal for the interactive sculpting of surfaces. Not only can the designer indirectly manipulate the surface by adjusting the degrees of freedom of the underlying geometry as is traditional in geometric modeling, but he can sculpt the dynamic surface directly, through the use of interactive sculpting tools in the form of simulated forces.

Fig. 11 illustrates four shapes sculpted using spring forces. The initial open surface is generated using a quadratic B-splines with 24 control points. Second, a cubic triangular planar patch with 10 control points shown in Fig. 12(a) was dynamically manipulated into the shape shown in Fig. 12(b).

7 Conclusion

We have proposed dynamic triangular NURBS, a new free-form shape model that marries the elegant geometry of rational multivariate simplex splines with physical dynamics. By using dynamic triangular NURBS, the designer can benefit from arbitrary parametric domains, non-degeneracy for multi-sided surfaces, and other important features.

We have applied Lagrangian dynamics to derive the equations of motion of dynamic triangular NURBS and techniques from finite element analysis to reduce the equations to efficient numerical algorithms. Our physics-based model responds to applied simulated forces with natural and predictable dynamics, while the

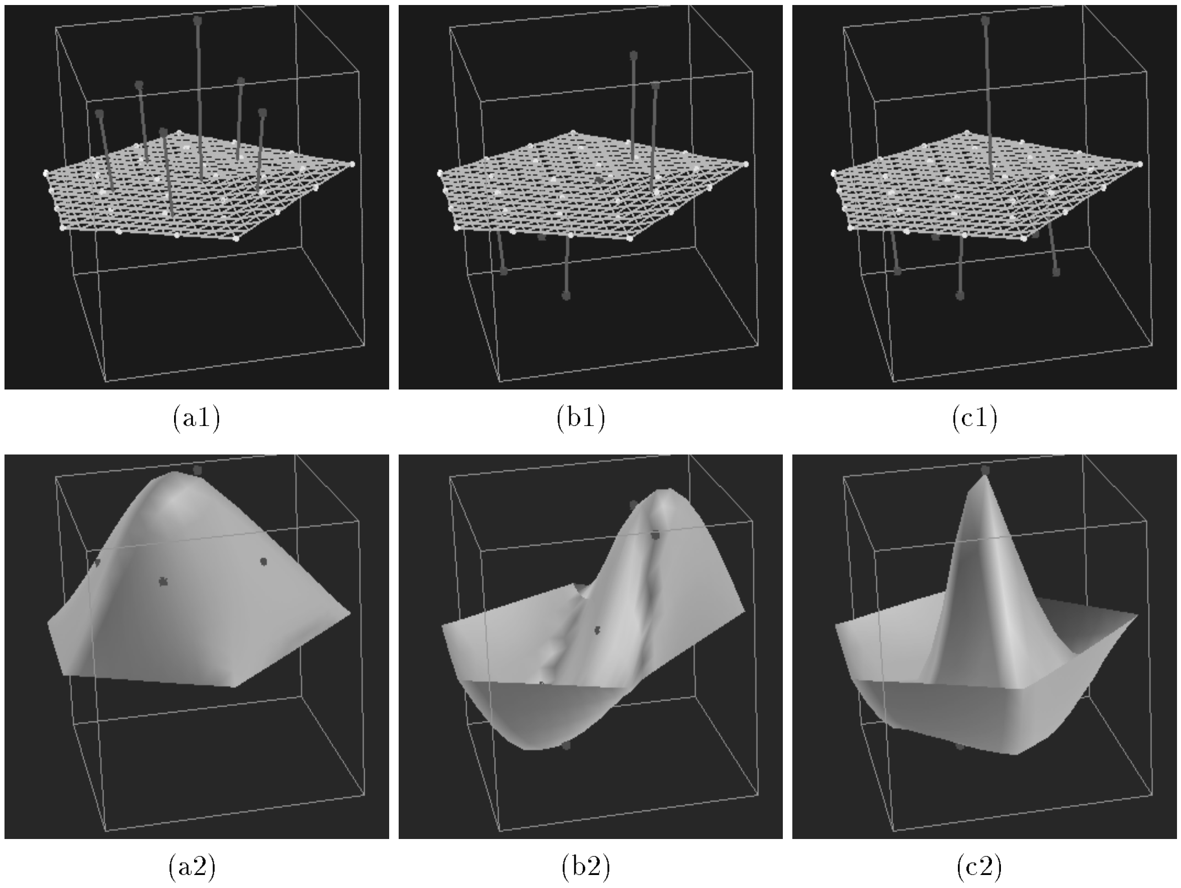


Figure 10: Fitting a pentagonal surface to three different sets of scattered data. Data points and initial surfaces (a1,b1,c1). Final fitted surfaces (a2,b2,c2).

underlying geometric parameters are determined automatically. Designers can employ force-based "tools" to manipulate the surface directly and interactively sculpt its shape. Additional control over the shape is available through the modification of physical parameters. Elastic energy functionals allow the qualitative imposition of fairness criteria through quantitative means. Linear or nonlinear constraints may be imposed either as hard constraints that must not be violated, or as soft constraints to be satisfied approximately in the form of simple forces. Constraint-based shape optimization is an automatic consequence of the dynamic model achieving static equilibrium.

Our experimental software demonstrates the ease of use of the special case of dynamic triangular B-splines in a variety of applications, including constraint-based optimization, automatic parametric design, shape blending, and interactive sculpting. Since our dynamic model is built upon existing geometric primitives, designers working with it can continue to use existing geometric design toolkits. Thus, dynamic triangular NURBS appear to be a promising new solid modeling tool.

Acknowledgments

We would like to thank Professor Hans-Peter Seidel of Universität Erlangen for kindly making available the software for triangular B-spline surfaces that he developed with Mr. Philip Fong. This research was made possible by grants from the Natural Sciences and Engineering Research Council of Canada and the Information Technology Research Center of Ontario. Terzopoulos is a fellow of the Canadian Institute for Advanced Research.

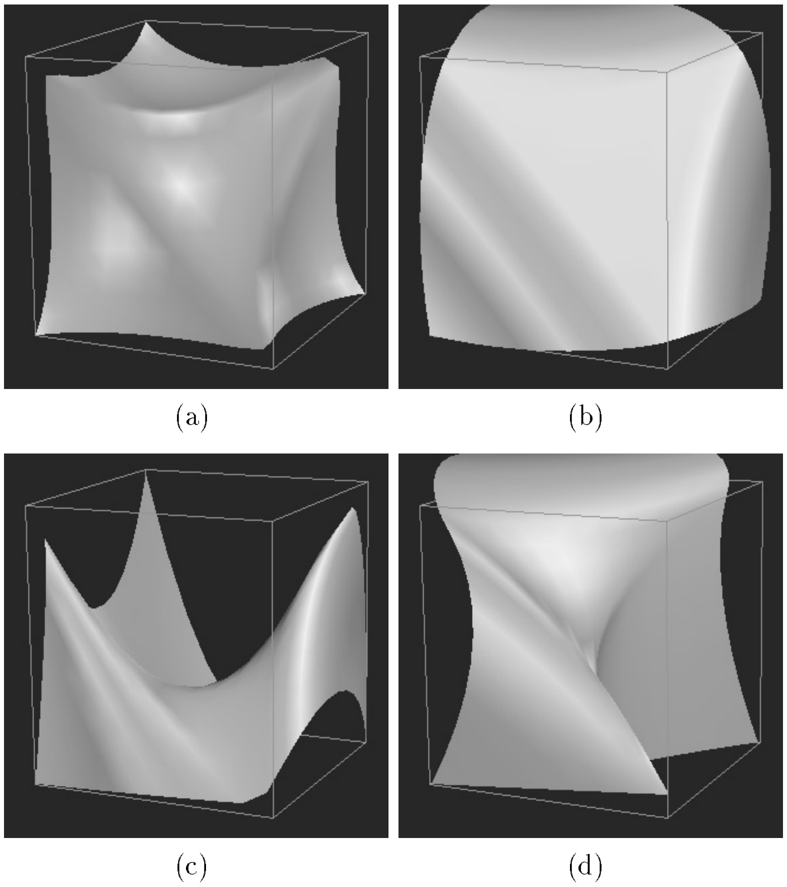


Figure 11: Interactive sculpting of an open quadratic surface into four different shapes (a–d).

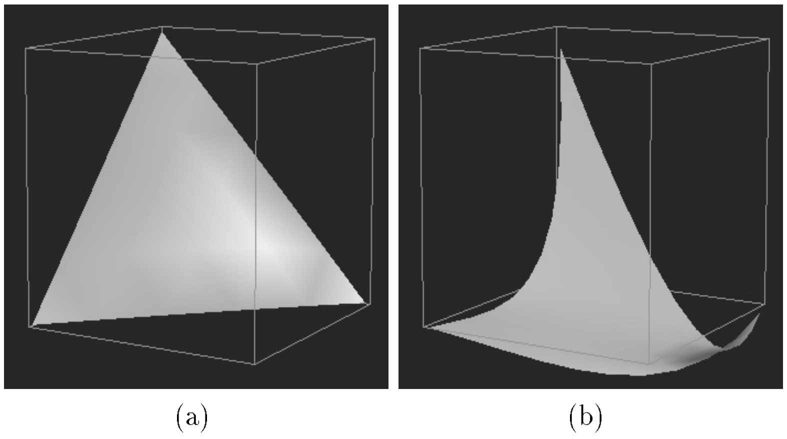


Figure 12: Interactive sculpting of a cubic patch (a) into another shape (b).

References

[1] S. Auerbach, R. Gmelig Meyling, M. Neamtu, and H. Schaeben. Approximation and geometric modeling with simplex B-splines associated with irregular triangles. *Computer Aided Geometric Design*, 8(1):67–87, 1991.

[2] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Comp. Meth. in Appl. Mech. and Eng.*, 1:1–16, 1972.

[3] M.I.G. Bloor and M.J. Wilson. Representing PDE surfaces in terms of B-splines. *Computer-Aided Design*, 22(6):324–331, 1990.

[4] G. Celniker and D. Gossard. Deformable curve and surface finite elements for free-form shape design. *Computer Graphics*, 25(4):257–266, 1991. (Proc. ACM Siggraph'91).

[5] W. Dahmen and C. Micchelli. On the linear independence of multivariate B-splines, I. triangulations of simploids. *SIAM J. Numer. Anal.*, 19(5):993–1012, 1982.

[6] W. Dahmen and C. Micchelli. Recent progress in multivariate splines. In C.K. Chui, L.L. Schumaker, and J.D. Ward, editors, *Approximation Theory IV*, pages 27–121. Academic Press, New York, 1983.

[7] W. Dahmen, C. Micchelli, and H.-P. Seidel. Blossoming begets B-spline bases built better by B-patches. *Mathematics of Computation*, 59(199):97–115, 1992.

[8] C. de Boor. Splines as linear combinations of B-splines. In G. Lorentz, C. Chui, and L.L. Schumaker, editors, *Approximation Theory II*, pages 1–47. Academic Press, New York, 1976.

[9] P. Fong and H.-P. Seidel. An implementation of triangular B-spline surfaces over arbitrary triangulations. *Computer Aided Geometric Design*, 3-4(10):267–275, 1993.

[10] B.R. Gossick. *Hamilton's Principle and Physical Systems.* Academic Press, New York and London, 1967.

[11] T. Grandine. The stable evaluation of multivariate simplex splines. *Mathematics of Computation*, 50(181):197–205, 1988.

[12] G. Greiner and H.-P. Seidel. Modeling with triangular B-splines. *IEEE Computer Graphics and Applications*, 14(2):56–60, 1994.

[13] K. Hollig. Multivariate splines. *SIAM J. Numer. Anal.*, 19(5):1013–1031, 1982.

[14] H. Kardestuncer. *Finite Element Handbook.* McGraw–Hill, New York, 1987.

[15] D. Metaxas and D. Terzopoulos. Dynamic deformation of solid primitives with constraints. *Computer Graphics*, 26(2):309–312, 1992. (Proc. ACM Siggraph'92).

[16] C.A. Micchelli. On a numerically efficient method for computing with multivariate B-splines. In W. Schempp and K. Zeller, editors, *Multivariate Approximation Theory*, pages 211–248. Birkhauser, Basel, 1979.

[17] M. Minoux. *Mathematical Programming.* Wiley, New York, 1986.

[18] H.P. Moreton and C.H. Sequin. Functional optimization for fair surface design. *Computer Graphics*, 26(2):167–176, 1992. (Proc. ACM Siggraph'92).

[19] R. Pfeifle and H.-P. Seidel. Fitting triangular B-Splines to functional scattered data. In *Proceedings of Graphics Interface'95*, pages 26–33, 1995.

[20] J. Platt. A generalization of dynamic constraints. *CVGIP: Graphical Models and Image Processing*, 54(6):516–525, 1992.

[21] W. Press, B. Flanney, S. Teukolsky, and W. Verttering. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, 1986.

[22] H. Qin and D. Terzopoulos. Dynamic manipulation of triangular B-splines. In *Proceedings of Third ACM/IEEE Symposium on Solid Modeling and Applications*, pages 351–360, Salt Lake City, May 1995. ACM Press.

[23] H. Qin and D. Terzopoulos. Dynamic NURBS swung surfaces for physics-based shape design. *Computer Aided Design*, 27(2):111–127, 1995.

[24] H.-P. Seidel. Representing piecewise polynomials as linear combinations of multivariate B-splines. In T. Lyche and L.L. Schumaker, editors, *Curves and Surfaces*, pages 559–566. Academic Press, New York, 1992.

[25] H.-P. Seidel. An introduction to polar forms. *IEEE Computer Graphics and Applications*, 13(1):38–46, 1993.

[26] D. Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(4):413–424, 1986.

[27] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, 1988.

[28] D. Terzopoulos and H. Qin. Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(2):103–136, 1994.

[29] C. Traas. Practice of bivariate simplicial splines. In W. Dahmen et al, editor, *Computation of Curves and Surfaces*, pages 383–422. Kluwer Academic Publishers, 1990.

[30] W. Welch and A. Witkin. Variational surface modeling. *Computer Graphics*, 26(2):157–166, 1992. (Proc. ACM Siggraph'92).